E0-245: ASP

Lecture 9: Complexity of Algorithms, Data Structures and Libraries: Part 1

Dipanjan Gope

Questions from Lecture 8

- Type erasure for generics

Module 1: OOPs and DS

JAVA and C++ mainly: (others C# and Objective-C)

- Object oriented programing: Classes and objects
- Inheritance, polymorphism, abstract class, const
- Templates and generics
- **Data structures**
- Standard library, JCF, STL
- **Complexity analysis**
- Multithreading and synchronization
- Good programming styles

Algorithm Complexity

Order of Complexity

1.2.1 Big Oh Notation

-

A convenient way of describing the growth rate of a function and hence the time complexity of an algorithm.

Let n be the size of the input and $f(n)$, $g(n)$ be positive functions of n.

DEF.Big Oh. $f(n)$ is $O(g(n))$ if and only if there exists a real, positive constant $C$and a positive integer n_0 such that

$$f(n) \leq Cg(n) \ \forall \ \ n \geq n_0$$

- Note that $O(g(n))$ is a class of functions.
- The "Oh" notation specifies asymptotic upper bounds
- $O(1)$ refers to constant time. $O(n)$ indicates linear time; $O(n^k)$ (k fixed) refers to polynomial time; $O(\log n)$ is called logarithmic time; $O(2^n)$ refers to exponential time, etc.

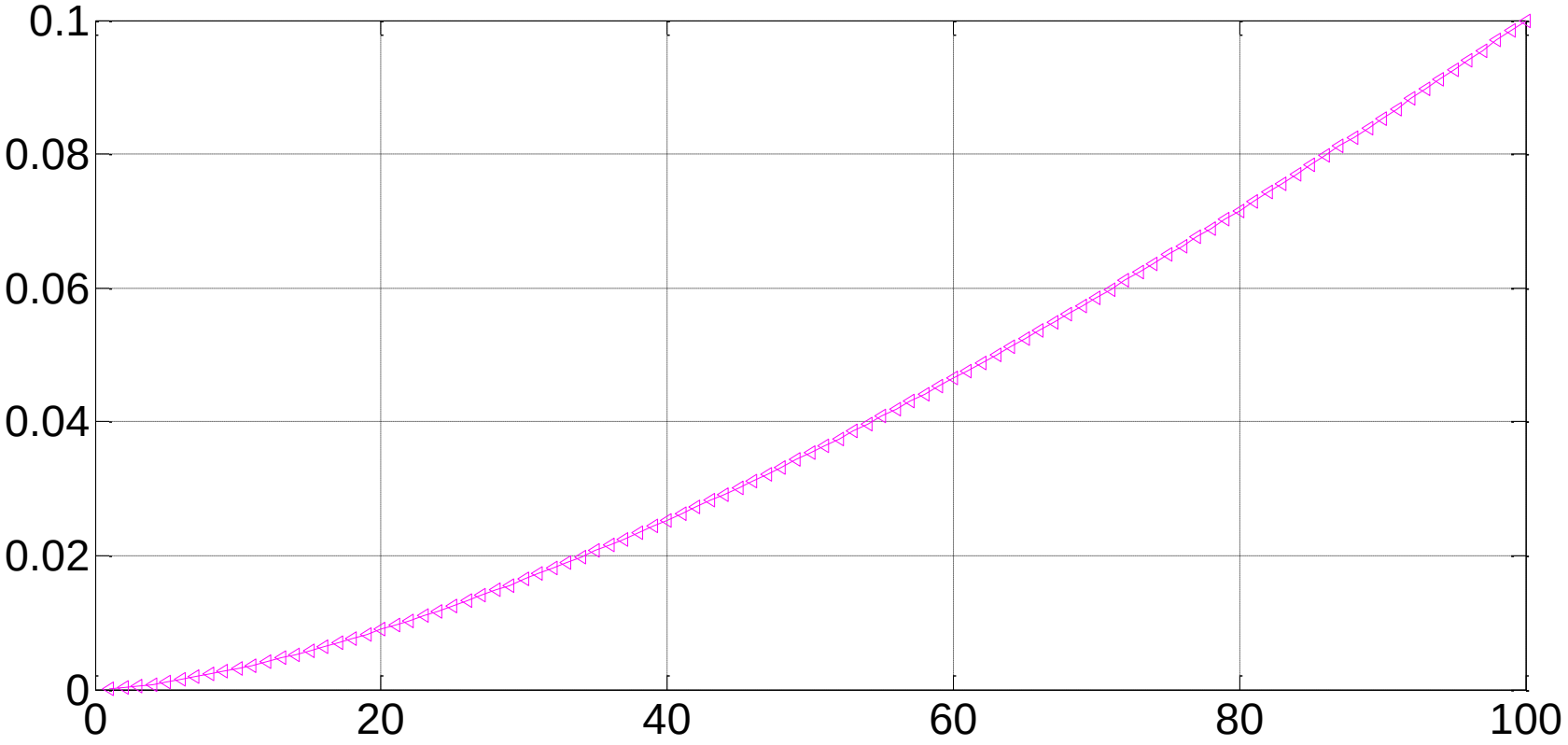
http://lcm.csa.iisc.ernet.in/dsa/node5.html: Prof. Narahari - CSA

Order of Complexity

- How computational time scales with number of inputs

  cost = constant * N^{γ}
- Polynomial complexity: $O(N^{\gamma})$
- Intractable complexity

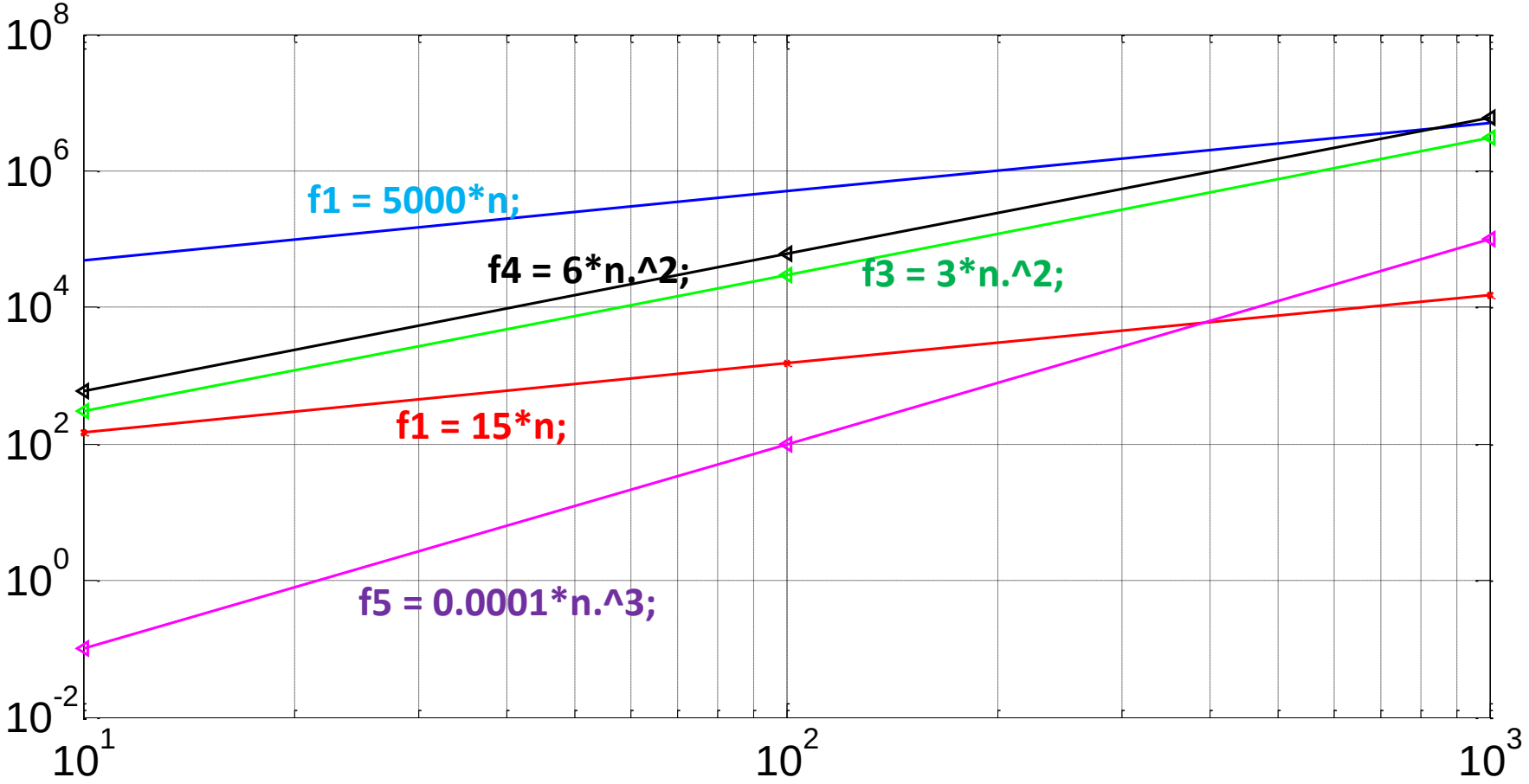
Find the complexity



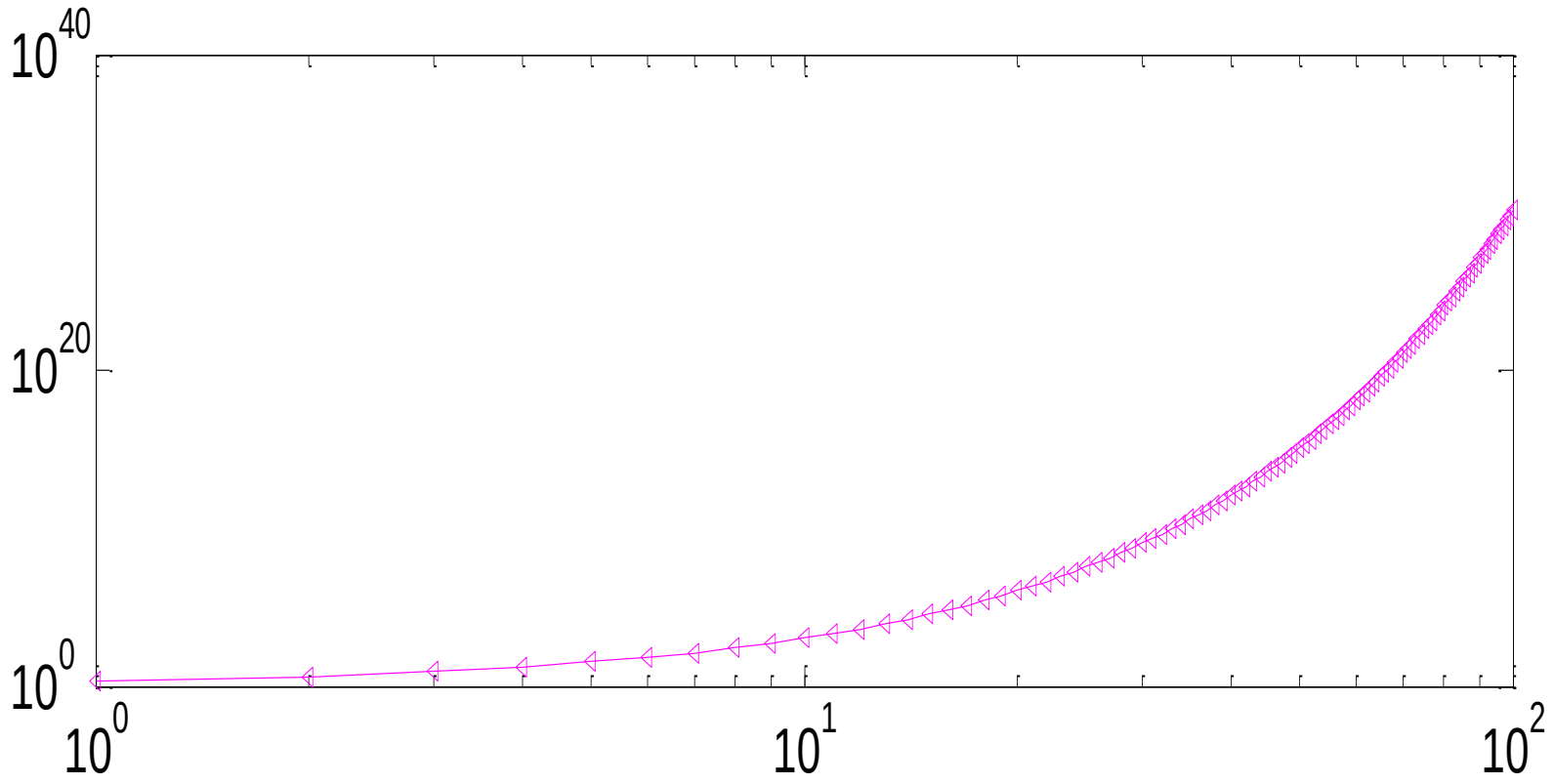
f=0.0001*n.^1.5;

Dipanjan Gope

Find the complexity



Find the complexity




```
f = 1*2.^n;
```

Compute the complexity

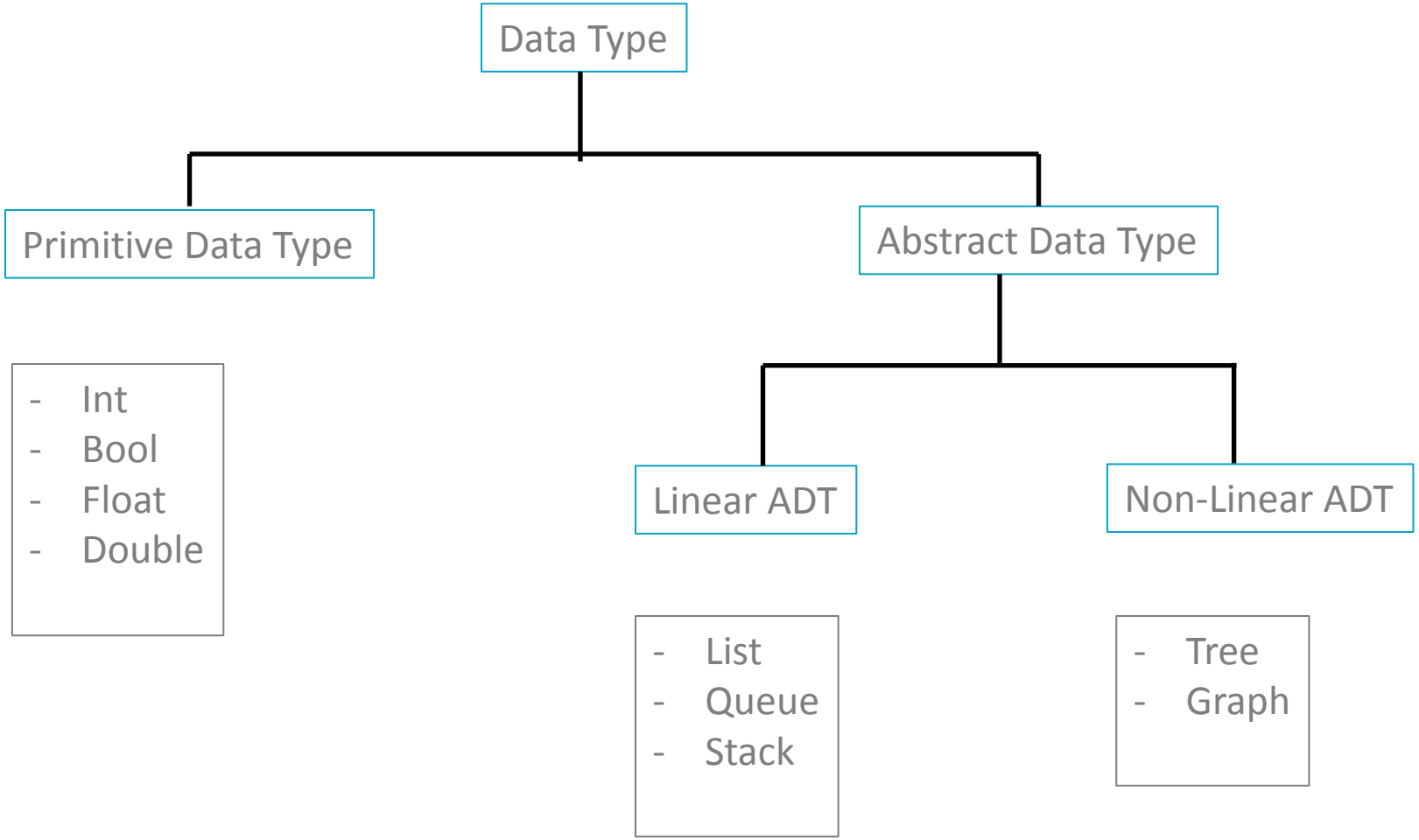
- Vector – Vector inner product
- Matrix – vector product
- Matrix – matrix product

Data Structures

Data Structures

- Organizing and storing data for efficient use
- An implementation of an Abstract Data Type
  - http://lcm.csa.iisc.ernet.in/dsa/node2.html Prof. Narahari (CSA)
- ADT Types:
  - Lists
  - Dictionaries
  - Trees
  - Graphs

Data Types

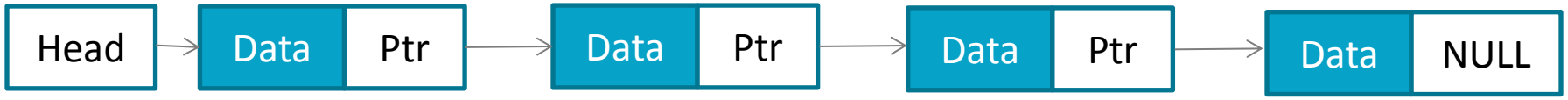

Data Type
Primitive Data Type
Abstract Data Type
- Int
- Bool
- Float
- Double
Linear ADT
Non-Linear ADT
- List
- Queue
- Stack
- Tree
- Graph

ADT List

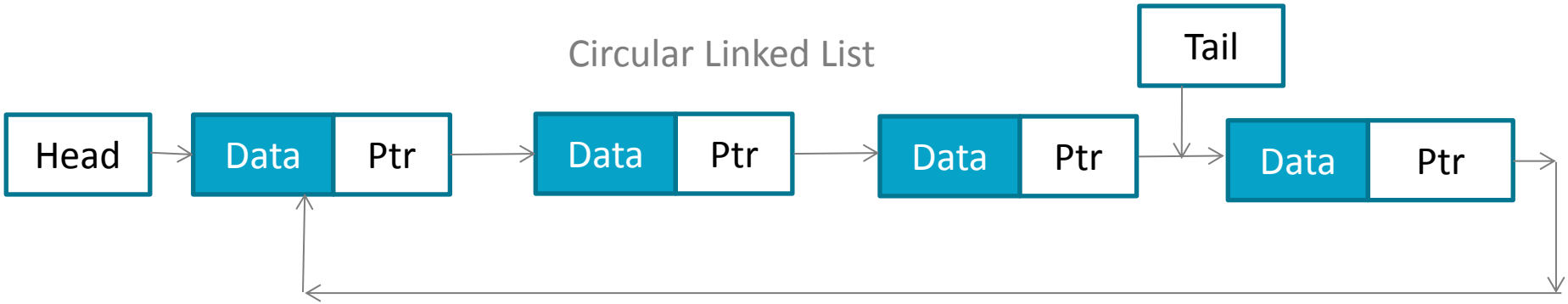
- Types: List, Stack, Queue
- Functions: access, add, remove, sort, search
- Implementation (DS): array, linked list

Linked List

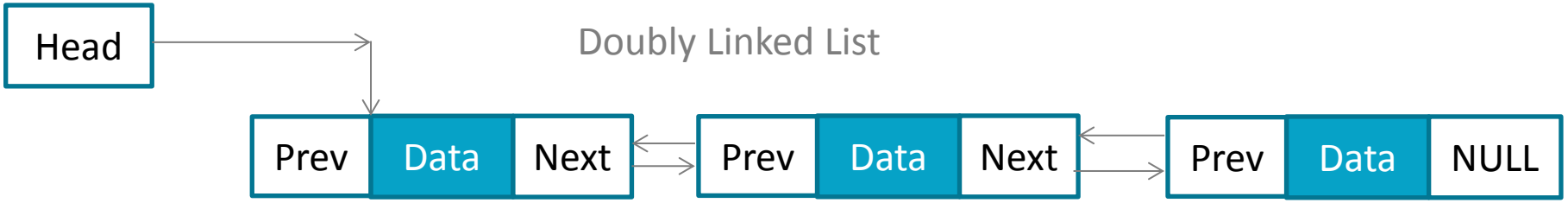
Singly Linked List



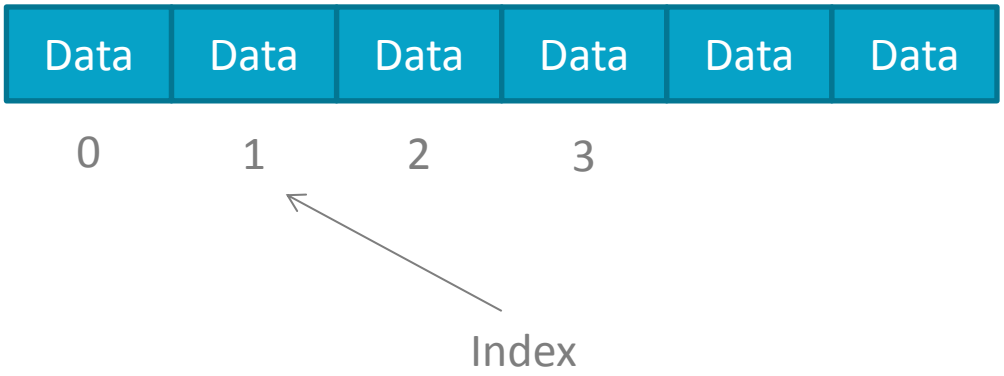
Circular Linked List



Doubly Linked List



Array



Array vs. Linked List

	Array	Linked List
Size	Fixed	Variable
Access	Quick	Lengthy
Deletion, Insertion	Tedious	Easy
Memory for fixed size	Lower	Higher

Algorithms on list: complexity

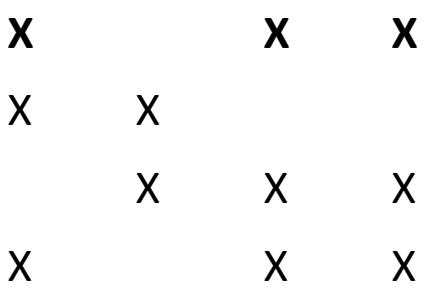
- Search
  - Naïve search (worst case complexity?)
  - Binary search (worst case complexity?)
- Sort
  - Quicksort (worst case complexity?)
  - Quicksort (average case complexity?)

Example: Dense Matrix Storage

X	**X**	**X**	**X**
X	X	X	X
X	X	X	X
X	X	X	X

- Array of arrays OR single array
- Row major / col major

Example: Sparse Matrix Storage



- i, j, val format
- i, val, rowStarts format (Yale Matrix Format)
- Compressed row or column